Towards a declarative method for 3D scene sketch modeling

Stéphane Donikian, Gérard Hégron

▶ **To cite this version:**

Stéphane Donikian, Gérard Hégron. Towards a declarative method for 3D scene sketch modeling. [Research Report] RR-1418, INRIA. 1991. inria-00075142

HAL Id: inria-00075142
https://hal.inria.fr/inria-00075142

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 1418

Programme 4
Robotique, Image et Vision

TOWARDS A DECLARATIVE METHOD FOR 3D SCENE SKETCH MODELING

Stéphane DONIKIAN
Gérard HEGRON

Avril 1991

RR-1418

IRISA INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTEMES ALEATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CEDEX
FRANCE
Téléphone : 99.36.20.00
Télex : UNIRISA 950 473F
Télécopie : 99.38.38.32

Towards a Declarative Method for 3D Scene Sketch Modeling

Stéphane DONIKIAN, Gérard HEGRON

IRISA/INRIA
Campus de Beaulieu
35042 Rennes cedex

Publication Interne n° 578 - mars 1991 - 22 pages

Programme 4

Abstract

This paper describes our first developments of a declarative method devoted to sketch creation of architectural environments. The architectural project conception we would try to simulate, is based on a script which describes the environment from a user's point of view who would move across the scene along trajectories. The objects are built by means of parallelepipedic boxes whose location and size are derived from logical rules (relative positionning of boxes) and from geometric constraints (on box sizes, distances, surface areas, volumes, etc.). A geometric instance of the scene is computed by using in a first step a logical inference which mainly takes into account the logical rules, and in a second step an optimization algorithm which deals with geometric constraints and provides a solution among the potential ones. The conception process which simulates the user's point of view going all over the environment is satisfied by a scene deformation technique.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(U.R.A. 227)
UNIVERSITE DE RENNES I I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(UNITE DE RECHERCHE DE RENNES)

Vers une méthode descriptive de modélisation d'esquisses de scènes tridimensionnelles

Résumé

Cet article présente nos premiers développements d'une méthode descriptive dédiée à la conception d'esquisses d'environnements architecturaux. La conception d'un projet architectural que nous tentons de simuler est basée sur un scripte qui décrit l'environnement du point de vue d'un usager qui se déplacerait à l'intérieur de la scène en suivant certaines trajectoires. Les objets sont ici créés à l'aide de boîtes parallèlépipédiques dont la taille et la position sont dérivées d'un ensemble de contraintes dites logiques (positionnement relatif des boîtes) et de contraintes géométriques reliant les dimensions des boîtes, leurs distances, leurs surfaces ou leurs volumes. Une configuration géométrique de la scène est calculée en utilisant dans une première étape une inférence logique qui prend en compte les contraintes logiques, et dans un second temps une technique d'optimisation qui intègre les contraintes géométriques et procure une solution parmi l'ensemble des configurations potentielles. Le processus de conception qui simule la perception d'un utilisateur parcourant l'environnement est satisfait à l'aide d'une technique de déformation de la scène.

1 Introduction

The general aim of this work is the implementation of a declarative method devoted to the description of 3D scenes. With respect to imperative methods, this approach permits to describe a scene by means of rules, properties or constraints from which a scene instance is computed by successive refinements. For a few years, this approach has been meeting a constantly increasing interest in all CAD domains [1, 2, 3]. We intend to apply this methodology to the conception of architectural environments. A few expert systems have been developed either to reproduce well known rules of construction [4] or to help the architect during the creation of his plans [5]. Our goal is not to get the final and precise geometric model of the scene, but to create sketches which meet the main properties and constraints of a mental image of the project

The architectural project conception we would try to simulate, is based on a script which describes the environment from a user's point of view who would move across the scene

along trajectories. In this paper we present our first developments of such a conception process. The objects are built by means of parallelepipedic boxes whose edges are parallel to the world coordinate system axes and whose location and sizes are derived from logical rules (relative positionning of boxes) and from geometric constraints (on box sizes, distances, surface areas, volumes, etc.). A geometric instance of the scene is computed by using in a first step the Allen's temporal logic [6] which mainly takes into account the logical rules, and in a second step an optimization algorithm which deals with geometric constraints and provides a solution among the potential ones. After having developed and illustrated how rules and constraints are managed, we show how we meet the conception process from a user point of view going all over the environment.

2 The design process

The scene generation process occurs in four steps:

1. **Scene description**
   The basic geometrical object is a rectangular parallelepiped. The user specifies the relative placement of the objects called logical constraints (like *above, below, behind, between*, and so on), and gives a set of geometrical constraints connecting object attributes (size, surface area, volume and so on). The use of default values and constraint disjunctions allows some degrees of freedom for the size and the relative placement of the objects.

2. **Logical resolution**
   A valuated graph is infered from the set of logical relations, which represents constraints between the beginning and end positions of box edges along the three axes. This step permits to check the logical consistency and to offer a first numerical approximation of the solution;

3. **Numerical resolution**
   The previous phase does not take geometrical constraints into account. The valuated graph is translated into an equivalent mechanical system (composed of points without mass connected by springs and/or abutments) whose stable position, found by minimizing its energy under geometrical constraints, gives a solution of the problem.

4. **Scene deformation**
   The scene is described during the first phase, as if it was seen from a straight trajectory. The scene pattern produced in this way is now subject to a deformation according to the trajectory shape.

The example below (fig. 2.1) shows an apartment. Most of relations give the object positioning with respect to its neighbours. The table is the only object which has degrees of freedom for its positioning (disjunction of relations) : it is constrained to be in the area

delimited by the two rooms on the left, the right wall, the front wall and the limit given by the back wall of the second room (see the top view).

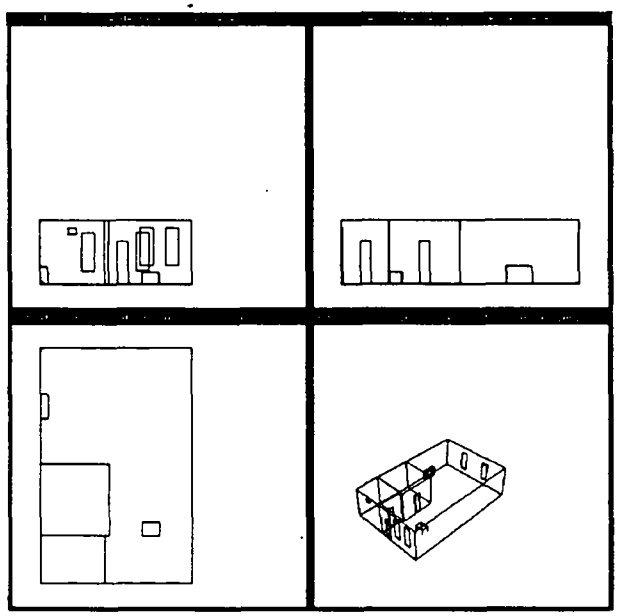


Figure 2.1 : Multiple views of the apartment

The interest in such a system is twofold. Firstly the sizes of solids and their relative positionning are generated in the same way, and secondly geometrical constraints are integrated. These constraints can be linear or non-linear. The use of abutments and springs in the mechanical model provides degrees of freedom to the system, even if sizes or tolerance intervals are imposed. The system is also able to give a new solution by modifying weighting coefficients on relations and constraints.

3 Logical resolution

As boxes with edges parallel to axis are used, a first modeling is performed by using the Allen's temporal logic [6] extended to the 3D space, where events represent space positions and temporal intervals the box edges. This model allows us to explicit spatial relations between objects along the three axes.

3.1 Allen's logic

The temporal interval is the basic entity manipulated by this logic. The duration is not explicitly represented and belongs to $]0, \infty[$. Thirteen relations are possible between two temporal intervals (fig. 3.1). These relations permit to describe every possible configuration between two intervals, and they are mutually disjunctive.

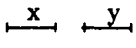
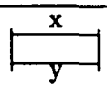
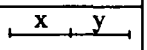
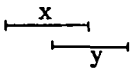
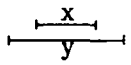
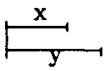
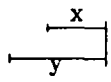
Relation	Symbol	Symbol for inverse	Illustration
x before y	<	>	
x equal y	=	=	
x meets y	m	mi	
x overlaps y	o	oi	
x during y	d	di	
x starts y	s	si	
x finishes y	f	fi	

Figure 3.1 : The thirteen temporal relations between two intervals

A constraint C between two intervals X and Y is a disjunction ($\vee$) of Allen's relation :

$C(X, Y) = X\,(\vee_{i=1}^{13} \alpha_i r_i)\,Y$ *with* $\alpha_i \in \{0, 1\}$ *and* r_i *one of the thirteen Allen's relation*

A lot of usual operations like intersection and composition are possible between constraints [7].

Each temporal interval can be represented by its beginning and end moments. Three relations are possible between these moments (precede, identical and follow). The deduction of relations on moments from these on intervals is a triviality. For example, Let $X = Y$ be the relation between two intervals X and Y. Let X_b and Y_b be the beginning moments of these intervals, and X_e and Y_e the end moments. The relations between moments are the following:

X_b (*identical to*) Y_b
X_b (*precede*) Y_e
X_e (*follow*) Y_b
X_e (*identical to*) Y_e

Afterwards, spatial segment takes the place of temporal interval and spatial point takes the place of temporal moment. A scene description is a conjunction ($\wedge$) of constraints which are themselves a disjunction of Allen's relations between segments. The "box on a table"

description (fig. 3.2) will be taken as example, in order to illustrate the design mechanism. Along each axis, we obtain the following expressions :

$$[(2s1) \wedge (4 = 2) \wedge (3f1) \wedge (5 = 3) \wedge (6d1)]\ along\ X\ axis$$

$$[(2m1) \wedge (3 = 2) \wedge (4 = 2) \wedge (5 = 2) \wedge (1m6)]\ along\ Y\ axis$$

$$[(2f1) \wedge (3 = 2) \wedge (4s1) \wedge (5 = 4) \wedge (6d1)]\ along\ Z\ axis$$

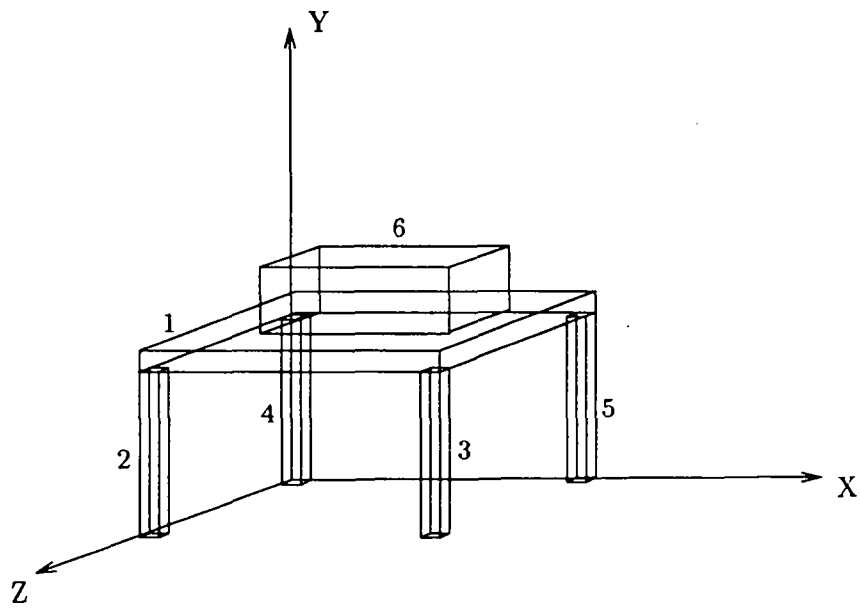


Figure 3.2 : Box on a table

3.2 Creation of a first valuated graph

The scene description given in Allen's logic is translated into a graph of segment extremities. The translation process is shown in this section. Each segment **X** is represented by its beginning and end points, which are two nodes $\mathbf{X_b}$ and $\mathbf{X_e}$ in the graph.

Allen's relation translation

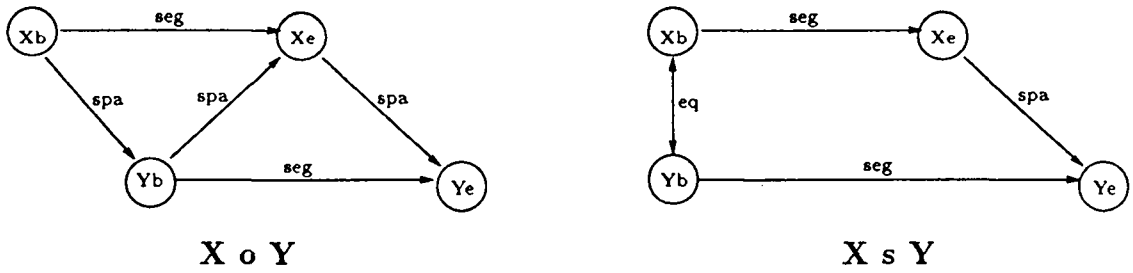
From each Allen's relation, a subgraph describing the links between the four nodes of the two involved segments is derived (see fig. 3.3). Each arc is valuated and three kinds of arc valuation can be derived :

$$X_b \xrightarrow{\textbf{seg}} X_e \iff X_b < X_e$$

$$X_k \xrightarrow{\textbf{spa}} Y_l \iff k,l \in \{b,e\} \wedge (X_k < Y_l) \wedge (X \neq Y)$$

$$X_k \xrightarrow{\textbf{eq}} Y_l \iff k,l \in \{b,e\} \wedge (X_k = Y_l) \wedge (X \neq Y)$$

The **seg** valuation is representing the relative positionning of the extremities of a spatial **segment**, while an **spa** valuation is representing the relative positionning of two extremities of two different spatial segments (spatial **spacing**), and an **eq** valuation is corresponding to an **equality** between two extremities of two different spatial segments.



X o Y **X s Y**

Figure 3.3 : Allen's relation translation examples

For the relation **X s Y**, there is not any arc between $\mathbf{Y_b}$ and $\mathbf{X_e}$ because, as the representation is made on an oriented axis, there is no need to represent an arc that could be deduced by transitivity (antitransitivity).

Constraint translation

A constraint is a disjunction of Allen's relations. There is only a subset of constraints which can be translated into an extremity graph, because of the translation of disjunction. A constraint translation is achieved by making the union of all extremity graphs corresponding to the translation of each Allen's relation composing the constraint. The union between two extremity graphs is possible, only if at the most one arc is different. Among all possible constraints (8191), there is only 187 which are compatible with the extremity graph representation [7]. This could seem to be an important restriction, but in fact it is not. The explanation is that constraints, compatible with extremity graphs, are corresponding to constraints which present a continuity in their geometrical configuration area. The two following examples are illustrating this compatibility.

The intersection relation between two segments is expressed by a disjunction of all relations where an intersection exists between segments (see fig. 3.4).

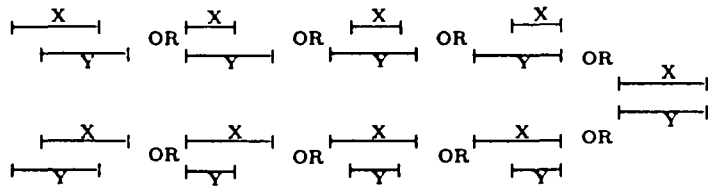


Figure 3.4 : The intersection relation

The intersection constraint will be written [X (o, o_i, d, d_i, s, s_i, f, f_i, =) Y]. We can use this constraint in the "box on the table" example, where it precisely means "a part of the box is on the table". The positionment between the box (6) and the upper part of the table (1) along X and Z axis is then [1 (o, o_i, d, d_i, s, s_i, f, f_i, =) 6]. The corresponding graph is shown below (fig. 3.5).

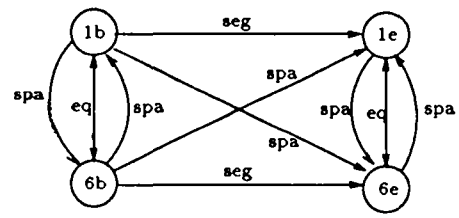


Figure 3.5 : Extremity graph of the "part of the box on the table" description along X axis.

The constraint [1 (o, o_i, =) 6] which is a part of the intersection constraint, should be represented by the disjunction of the three graphs (fig. 3.6). The union of the three graphs should give the graph of the figure 3.5, but the constraint is not compatible with the resulting extremity graph.

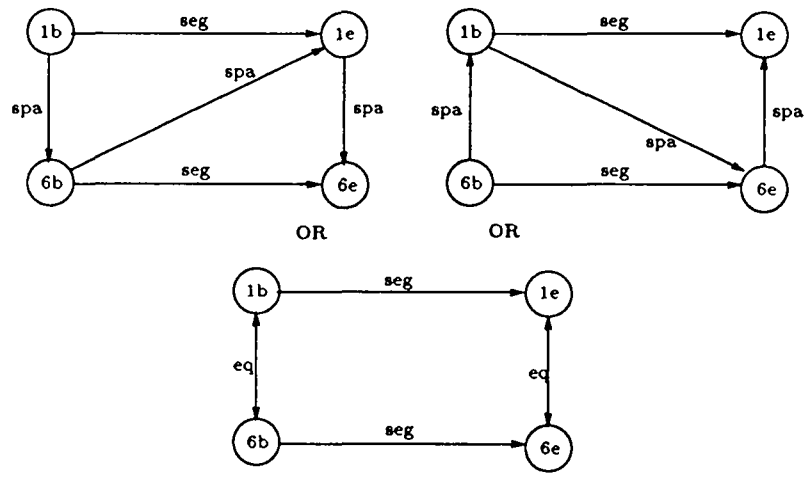


Figure 3.6 : Extremity graphs of the constraint [1 (o, o_i, =) 6].

description translation

A description beeing a conjunction of constraints, is expressed by making the union of graphs corresponding to each constraint. This description is logically consistent if there is at least one solution to order extremities of segments. For example, the set of constraints (B d A), (A m C), (B o C) is logically inconsistent, because points A_e, B_e and C_b cannot be ordered ($A_e \overset{eq}{\leftrightarrow} C_b \overset{spa}{\rightarrow} B_e \overset{spa}{\rightarrow} A_e$).

Figure 3.7 illustrates the valuated graph of the "box on a table" example, where nodes $\mathbf{0_b}$ and $\mathbf{0_e}$ correspond to extremities of the "box on a table" description domain.

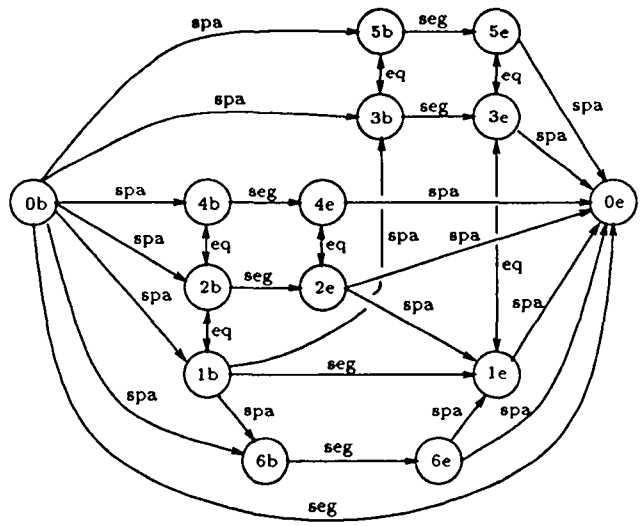


Figure 3.7 : Extremity graph of the "box on a table" description along X axis.

3.3 Validation and estimation of the model for each axis

After the graph corresponding to the set of logical constraints is obtained, a validation and estimation algorithm is applied to the model. Its aim is twofold: on the one hand there is a logical consistency test (no circuit in the extremity graph) and on the other hand a new graph is obtained by minimizing the extremity graph and integrating numerical data. The algorithm occurs in four steps.

Phase 1 : obtaining of the new graph

The new graph is obtained by the ordered use of the following eight rules.

1. $\forall X,Y.(X_k \overset{spa}{\rightarrow} Y_l) \wedge (X_k \overset{eq}{\rightarrow} Y_l) \wedge (Y_l \overset{spa}{\rightarrow} X_k) \wedge nomark(X_k \overset{spa}{\rightarrow} Y_l, X_k \overset{eq}{\rightarrow} Y_l, Y_l \overset{spa}{\rightarrow} X_k) \Rightarrow mark(X_k \overset{spa}{\rightarrow} Y_l, X_k \overset{eq}{\rightarrow} Y_l, Y_l \overset{spa}{\rightarrow} X_k)$

2. $\forall X,Y.(X_k \overset{spa}{\rightarrow} Y_l) \wedge (Y_l \overset{spa}{\rightarrow} X_k) \wedge nomark(X_k \overset{spa}{\rightarrow} Y_l, Y_l \overset{spa}{\rightarrow} X_k) \Rightarrow mark(X_k \overset{spa}{\rightarrow} Y_l, Y_l \overset{spa}{\rightarrow} X_k)$

3. $\forall X, Y.\,(X_k \overset{\text{spa}}{\rightarrow} 0_e) \wedge (X_k \overset{\text{eq}}{\rightarrow} 0_e) \wedge nomark(X_k \overset{\text{spa}}{\rightarrow} 0_e)$
   $\Rightarrow mark(X_k \overset{\text{spa}}{\rightarrow} 0_e)$

4. $\forall X, Y.\,(0_b \overset{\text{spa}}{\rightarrow} X_k) \wedge (0_b \overset{\text{eq}}{\rightarrow} X_k) \wedge nomark(0_b \overset{\text{spa}}{\rightarrow} X_k)$
   $\Rightarrow mark(0_b \overset{\text{spa}}{\rightarrow} X_k)$

5. $\forall X, Y.\,(X_k \overset{\text{eq}}{\rightarrow} Y_l) \wedge \neg(X_k \overset{\text{spa}}{\rightarrow} Y_l) \wedge \neg(Y_l \overset{\text{spa}}{\rightarrow} X_k) \wedge nomark(X_k \overset{\text{eq}}{\rightarrow} Y_l)$
   $\Rightarrow equality(X_k, Y_l), mark(X_k \overset{\text{eq}}{\rightarrow} Y_l)$

6. $\forall X, Y.\,(X_k \overset{\text{eq}}{\rightarrow} Y_l) \wedge (X_k \overset{\text{spa}}{\rightarrow} Y_l) \wedge nomark(X_k \overset{\text{spa}}{\rightarrow} Y_l, X_k \overset{\text{eq}}{\rightarrow} Y_l)$
   $\Rightarrow add(X_k \overset{\text{less-eq}}{\rightarrow} Y_l), mark(X_k \overset{\text{spa}}{\rightarrow} Y_l, X_k \overset{\text{eq}}{\rightarrow} Y_l)$

7. $\forall X, Y.\,(X_k \overset{\text{spa}}{\rightarrow} Y_l) \wedge nomark(X_k \overset{\text{spa}}{\rightarrow} Y_l)$
   $\Rightarrow add(X_k \overset{\text{less-spa}}{\rightarrow} Y_l), mark(X_k \overset{\text{spa}}{\rightarrow} Y_l)$

8. $\forall X, Y.\,(X_k \overset{\text{seg}}{\rightarrow} Y_l) \wedge nomark(X_k \overset{\text{seg}}{\rightarrow} Y_l)$
   $\Rightarrow add(X_k \overset{\text{less-seg}}{\rightarrow} Y_l), mark(X_k \overset{\text{seg}}{\rightarrow} Y_l)$

The *mark* function produces the marking of all arcs in its argument, to forbid a second use of these arcs. The *nomark* function checks that any arc in its argument is already marked. The *equality* function is used to minimize the arc number, by grouping together all equal nodes. The *add* function produces the addition of an arc in the new graph.

Rule 1 and 2 expresses that there is no creation of an arc connecting two nodes when the disjunctive relations are $(<,=,>)$ and $(<,>)$. Rule 3 and 4 expresses contact with the domain extremities. Rules 5 to 8 are only rewriting rules. The **less-eq** valuation (rule 6) corresponds to the *"less or equal"* relation between two extremities of different segments. The **less-spa** valuation (rule 7) corresponds to the *"strictly less"* relation between two extremities of different segments. As for the **less-seg** valuation (rule 8), it corresponds to the *"strictly less"* relation between the extremities of the same segment. There is a fourth valuation (**equal**), which is obtained by applying the *equality* function (rule 5).

Phase 2 : verification of the logical consistency and construction of node attributes

Nodes attributes (list of successors, index corresponding to the maximal length path on the left of the node in terms of arc number) are built from an exhaustive and single graph traversal. During this traversal, a logical inconsistency is detected when a circuit is found in the graph.

Phase 3 : partial antitransitivity

For each arc from X_k to Y_l which is not linked by a geometrical constraint, if there is a path between its two nodes of length higher than one, then the arc is removed applying

antitransitivity rule. Figure 3.8 shows the new "part of the box on a table" graph obtained after the execution of phase three.

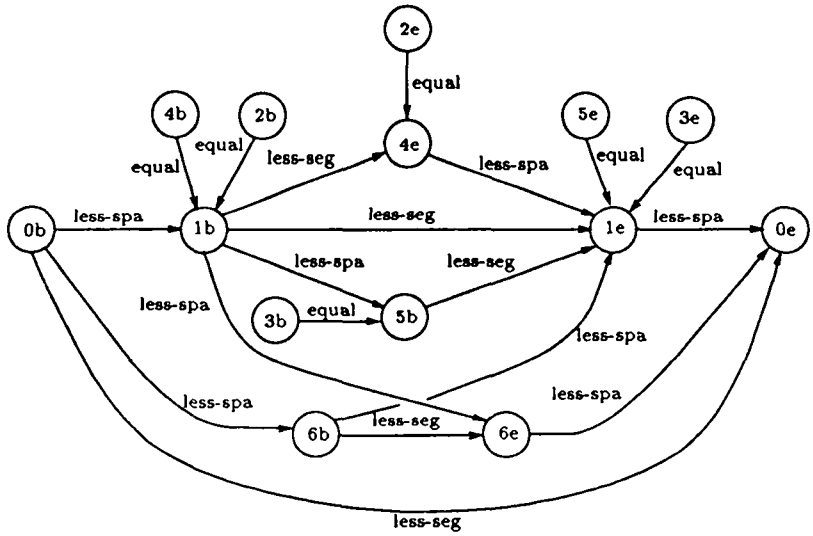


Figure 3.8 : New "part of the box on a table" graph along X axis.

Phase 4 : estimation of unknown values

In order to lighten the paper content, geometrical constraints have not been mentioned yet. These constraints can be of two types:

1. Monodimensional constraints (linear) on distance between two points: length of an object, distance or intersection between two segments. This distance can be a fixed or bounded value. It will be represented by a medium value and its two extrema (minimum and maximum values).

2. Multidimensional constraints (nonlinear) between points, lines, surfaces and volumes : these constraints are represented by their corresponding analytical equations.

At the first stage, all existing monodimensional constraints are integrated into the new graph, and an algorithm is applied to estimate unknown lengths (see appendix). At the second stage, multidimensional constraint equations are stored and will be used for the mechanical system resolution.

3.4 Numerical resolution

The translation of the problem into a mechanical system permits to solve simultaneously geometrical and logical constraints and to propose one of possible solutions (an infinity) by computing an equilibrium state of the system. Each arc from $\mathbf{X_k}$ to $\mathbf{Y_l}$ corresponds to a spring with the following characteristics:

- a rest length $\mathbf{l_{0(X_k,Y_l)}}$ which is the given or estimated arc length ;
- a stiffness coefficient $\mathbf{k_{(X_k,Y_l)}}$ depending on possible values of $(Y_l - X_k)$;
- two compressibility and extensibility bounds $\mathbf{l_{min(X_k,Y_l)}}$ and $\mathbf{l_{max(X_k,Y_l)}}$ represented by abutments.

The mechanical system is represented by binding equations (abutments and nonlinear constraints) and work of springs. The solution of the system is performed by using a penalty method [8]. The equation system to solve is the following:

$$\begin{aligned} &\mathcal{Q}_k + \alpha \sum_h f_h \frac{\partial f_h}{\partial q_k} = 0, \quad k = 1, 2, ..., n \\ &with\, b_j(q) \leq 0, \qquad\qquad\quad j = 1, 2, ..., p \end{aligned} \tag{1}$$

where

- $q = (q_k)_{k=1,n}$ is the set of lagrangian parameters which are here segment extremities (graph nodes) ;
- $\mathcal{Q}_k$ is the generalized given effect relative to q_k, here the spring forces, in terms of $k.(l_{0(X_k,Y_l)} - (Y_l - X_k))$;
- f_h are the bilateral constraints. They are the analytical equations of the multidimensional geometric constraints ;
- b_k represents the unilateral constraints coming from abutments : $(l_{min(X_k,Y_l)} - (Y_l - X_k) \leq 0)$ and $((Y_l - X_k) - l_{max(X_k,Y_l)} \leq 0)$;
- α is a chosen penalty constant.

According to the kind of wished solution, the user can bend the suggested solution by inflecting constraint weighting coefficients : spring stiffness coefficients to fix or to relax segment length, weight of a geometrical constraint to modify its priority.

4 Management of constraints : an example

An example is shown to illustrate how constraints can be managed during scene design process. In this example, the scene is composed of three objects : a table, a box and a window (see fig. 4.1). Initial constraints are the following : *"the table is on the floor"*, *"a part of the box is on the table"*, *"the window is on the back wall"*.

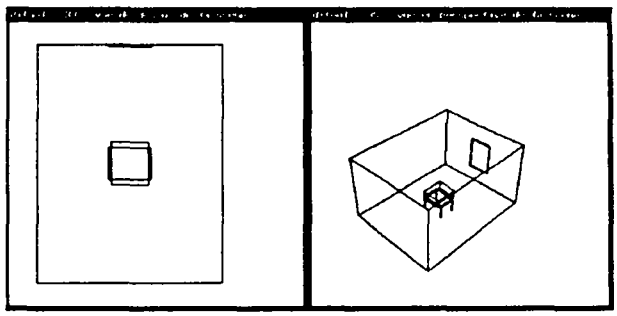


Figure 4.1 : Initial scene

The scene 2 is obtained (fig. 4.2) after the addition of the following constraint *"The table is to the left of the window"*.

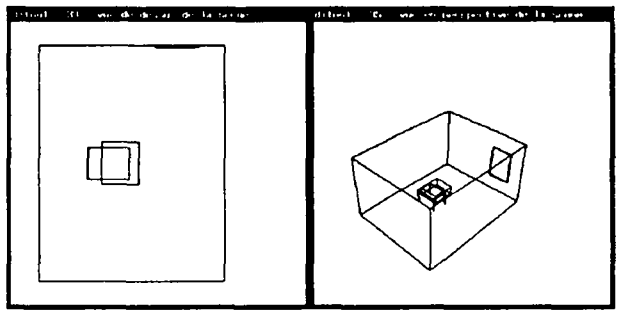


Figure 4.2 : Scene 2

The window having never been constrainted along the X axis, moves to the right. If a user adds the following constraint *"the window is in the middle of the back wall"* then figure 4.3 is obtained.

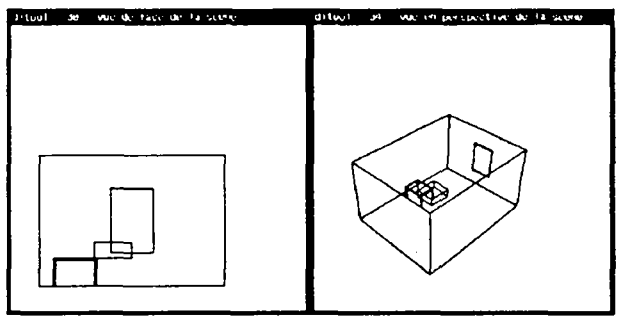


Figure 4.3 : Scene 3

The box is on the table but precariously balanced on the edge. If an equilibrium constraint is added (*"more than half the surface of the box is on the table"*), then figure 4.4 is obtained.

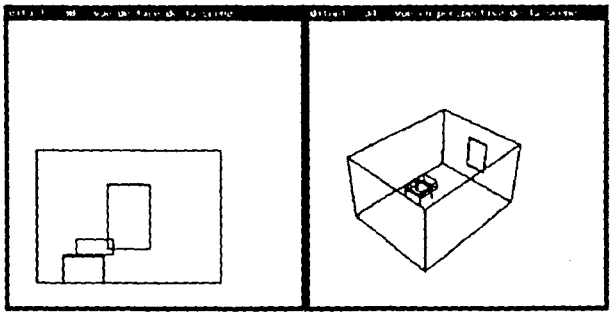


Figure 4.4 : Scene 4

The addition of the constraint *"the box is inside the window"* produces a logical inconsistency in the graph of constraints. In the same way, the addition of the constraint *"The box is in the middle of the scene"* produces a numerical inconsistency. This remark points out another interest of using the mechanical system which permits to simply take into account both logical and geometrical constraints. Detecting such an inconsistency with an expert system should require to integrate geometrical constraints and associated rules, and then to infer on this set of knowledge. With this solver, the user is able to know the set of transgressed constraints, and then to inflect them.

5 Scene deformation

During the architectural project conception, we try to simulate the perception of a user who moves across the scene. The scene description is incremental. The user introduces objects and constraints as he advances. The description model permits to define relative placement of objects as if they were perceived (orthographic projection) along a rectilinear trajectory which is a kind of scene skeleton. In the previous scene example (fig. 4.1), the skeleton could be a straight line from the middle of the front wall to the middle of the back wall. In a way, the solution is a *normalized* representation of the scene. Using techniques of object deformation and space reconstruction from multiple points of view, more complex scenes can be generated simply. For example, the Stonehenge's stone placement (fig. 5.1) is designed as an alignment along a rectilinear trajectory ; The final scene is then obtained by projecting the normalized space onto the curved space described by the trajectory (skeleton) from which the observer describes the objects (deformation process). The scene would have been described by a user walking along the alignment. Therefore the trajectory would be a circle. The alignment is here perceived as if the observer was located in the middle of the scene and looked at the stones turning round on himself.

A prototype has been written in prologII, and at present, scenes are described in this language. Let us detail the Stonehenge example. The alignment of pillars and beams is defined by an iterative process and the corresponding prologII predicate is the following :

```
alignment  ->
   domain(1000,100,300)
   pillar-beam(10,"0");


pillar-beam(0,t)  ->
   rel-ending(0,t)
   /;
pillar-beam(i,"0")  ->
   val(sub(i,1),i')
   pillar(<u-1,j>)
   beam(<u-2,k>)
   rel-initial(j,k)
   pillar-beam(i',k)
   /;
pillar-beam(i,p)  ->
   val(sub(i,1),i')
   pillar(<u-1,j>)
   beam(<u-2,k>)
   rel-iteration(j,k,p)
   pillar-beam(i',k)
   /;
```

```
rel-ending(0,t)  ->
  ff(t,"0","x");


rel-initial(j,k)  ->
   oo(j,k,"x")
   mm(j,k,"y")
   equal(j,k,"z")
   ss(j,"0","z")
   ss(j,"0","y")
   ss(j,"0","x");

rel-iteration(j,k,p)  ->
   oo(j,k,"x")
   mm(j,k,"y")
   equal(j,k,"z")
   ss(j,"0","z")
   ss(j,"0","y")
   mm(p,k,"x")
   oo(p,j,"x");
```

The iteration main part is standard and corresponds to an initialization, a loop and an ending condition. The predicate set can be automatically obtained from the knowledge on objects and their relations.

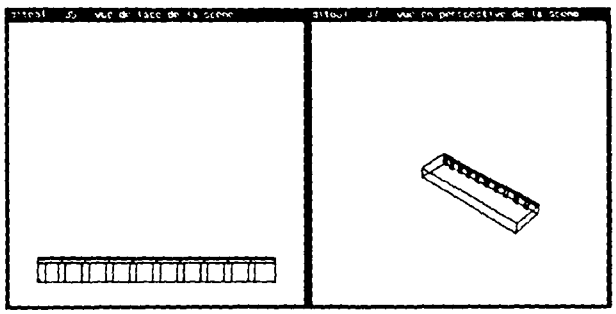


Figure 5.1 : Alignment as if it was described from a rectilinear trajectory

In order to take into account the trajectory from which the objects are described, a space deformation function (fig. 5.2) is applied on the *normalized* representation of the scene. Here is the function definition for Stonehenge example :

$$\begin{array}{ccc} \Re^3 & \rightarrow & \Re^3 \\ \begin{pmatrix} x \\ y \\ z \end{pmatrix} & \Rightarrow & \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \end{array} \quad / \quad \begin{cases} x' = \frac{l_x}{2\pi}(1 + (1 - \frac{z}{l_z}) * cos(\frac{x}{l_x} * 2\pi)) \\ y' = y \\ z' = \frac{l_x}{2\pi}(1 + (1 - \frac{z}{l_z}) * sin(\frac{x}{l_x} * 2\pi)) \end{cases}$$

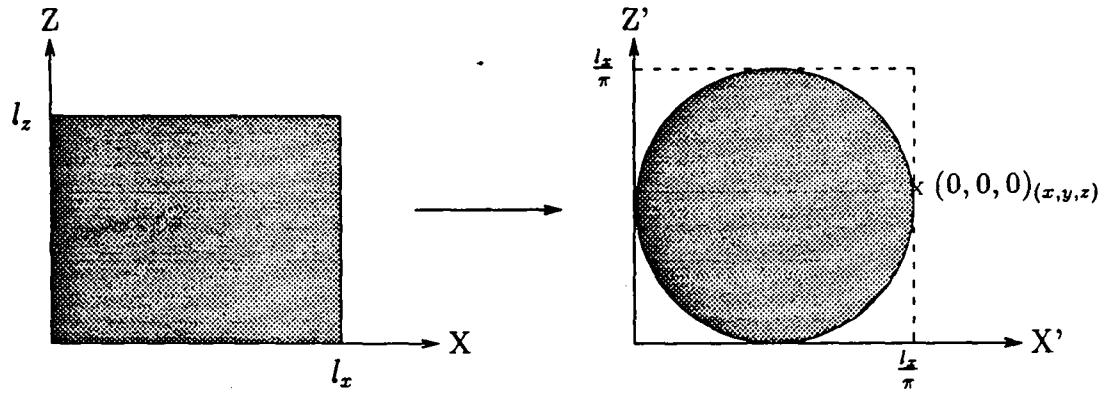


Figure 5.2 : Scene deformation function.

In this deformation, images of points $(0, y, l_z)_{(x,y,z)}$ and $(l_x, y, l_z)_{(x,y,z)}$ meet at the same point $(\frac{l_x}{2\pi}, y, \frac{l_x}{2\pi})_{(x',y',z')}$. In the same way, the image of points $(0, y, 0)_{(x,y,z)}$ and $(l_x, y, 0)_{(x,y,z)}$ is $(\frac{l_x}{\pi}, y, \frac{l_x}{2\pi})_{(x',y',z')}$. Finally, after this deformation the Stonehenge sketch is obtained (fig. 5.3).

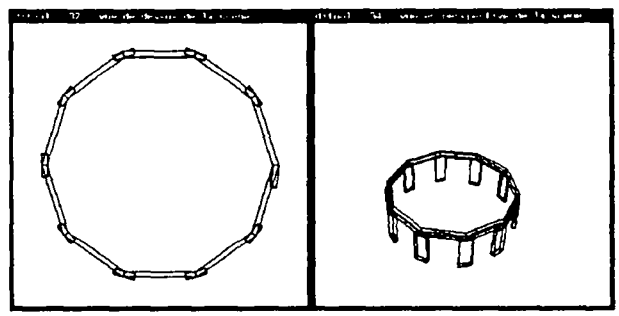


Figure 5.3 : Stonehenge sketch.

6 Conclusion

The paper presents the first steps in the implementation of our declarative conception of architectural environments. One of its main characteristics is the cooperation between two inference techniques which are most of the time competing : on the one hand a logical inference stemming from Allen's temporal logic, on the other hand a numerical inference based on an optimization under constraints algorithm.

An object oriented formalism is being investigated to structure the knowledge. This formalism will integrate architectural rules and attributes. Relying on this knowledge representation a declarative description of path in the scene will be explored as well as the generalization of the scene deformation process along trajectories. The implementation of a natural description language might be the final stage of this work.

Acknowledgements

We would like to thank B. Arnaldi and G. Dumont for their contribution to the numerical solver and the team of the architectural school of Rennes (E.A.B) for their fruitful discussions.

References

[1] P.J.W. ten Hagen and T. Tomiyama (Eds.). *Intelligent CAD Systems I. Theoretical and methodological aspects.* Springer-Verlag, 1987.

[2] D. and P. Martin. An expert system for polyhedra modelling. In *EUROGRAPHICS'88 Conference Proceedings*, 1988.

[3] C. Colin. A system for exploring the universe of polyhedral shapes. In *EUROGRAPHICS'88 Conference Proceedings*, 1988.

[4] S. Bessette, J. Vaucher, M. Fregier, and N. Chourot. Système expert pour la composition de colonnades d'ordres classiques selon les règles de l'architecture régulière formulées par andréa palladio. *CAD and Robotics in architecture and construction*, 300–323, 1987.

[5] U. Flemming, R.F. Coyne, T. Glavin, H. Hsi, and M. Rychener. *A Generative Expert System for the Design of Building Layouts(Final Report).* Technical Report EDRC 48-15-89, Carnegy Mellon University, 1989.

[6] J.F. Allen. An interval based representation of temporal knowledge. In *Proceedings of the seventh International Joint Conference on Artificial Intelligence*, pages 221–226, August 1981.

[7] T. Granier. *Contribution à l'étude du temps objectif dans le raisonnement.* Technical Report 716-I-73, IMAG, Février 1988.

[8] B. Arnaldi, G. Dumont, and G. Hégron. Animation of physical systems from geometric kinematic and dynamic models. In *Proc. of Working Conference on Modeling in Computer Graphics*, IFIP TC 5/WG 5.10, to appear, 1991.

APPENDIX

Unknown length estimation algorithm

For each arc from X_k to Y_l of unknown length, this algorithm estimates the $d(X_k, Y_l)$ distance. This distance is estimated with aid of *maxl* and *maxr* maximal path length functions. The *maxl*(X_k) function corresponds to the maximal distance between the beginning node 0_b and a node X_k. In the same way *maxr*(X_k) function corresponds to the maximal distance

between a node X_k and the end node 0_e. For the X_k node, those distances are evaluated by the following recursive functions :

- $\text{maxl(i)} = \max_{u \in pred_1(i)}(\text{maxl(u)} + \text{d(u,i)})$
- $\text{maxr(i)} = \max_{u \in succ_1(i)}(\text{maxr(u)} + \text{d(i,u)})$

where $pred_1(i)$ (respectively $succ_1(i)$) is the direct predecessor set (respectively successor) of the i node.

Here is the unformal description of the algorithm for each axis :

1. evaluation of initial *maxl* and *maxr* values for each graph node, with respect to the fixed data given by the user ;
2. let n be the index of 0_f. The evaluation of each d(i, j) is obtained by the execution of *calculation-lr(0,n)*.

The following *calculation-lr* (respectively *calculation-rl*) function calculates *d(i, j)* by a graph traversal from left to right (respectively from right to left). Those functions are recursively crossed in order to balance the effect of left and right maximum length pathes. Only one step of the recursivity is applied because a rough estimation is sufficient to give valid inputs to the mechanical system.

```
proc calculation-lr(k,l);
 int x,y,m;
 begin
  if (k ≥ l) then exit fif
  for all x with index k do
  for all y ∈ succ1(x) do
   if not(eval(x,y)) then
     m := MED(maxl(x),maxl(y),maxr(x),maxr(y));
     else d(x,y) := m;
     endif
   endif
  done
  done
 calculation-rl(k,l);
 end
```

```
proc calculation-rl(k,l);
 int x,y,m,k',l';
 begin
 if (k ≠ l) then
  for all y with index l do
  for all x ∈ pred1(y) do
   if not(eval(x,y)) then
     m := MED(maxl(x),maxl(y),maxr(x),maxr(y));
     else d(x,y) := m;
     endif
   endif
  done
  done
 k' := k+1;
 l' := l-1;
 calculation-lr(k',l');
 endif
 end
```

The *MED* function calculates a medium value of *d(x,y)* according to *maxl* and *maxr* functions and to free space :

if (maxl(x) = maxl(y)) and (maxr(x) = maxr(y))
then MED(maxl(x), maxl(y), maxr(x), maxr(y)) = max(0,d(0d,0f) - (maxl(x)+maxr(x))/2) ;
else MED(maxl(x), maxl(y), maxr(x), maxr(y)) = max(0,[(maxl(y)-maxl(x)) + (maxr(x)-maxr(y))]/2) ;

LISTE DES DERNIERES PUBLICATIONS INTERNES IRISA

1991

PI 570 DESIGN DECISION FOR THE FTM : A GENERAL PURPOSE FAUTL TOLERANT MACHINE
Michel BANATRE, Gilles MULLER, Bruno ROCHAT, Patrick SANCHEZ
Janvier 1991, 30 pages

PI 571 ANIMATION CONTROLEE PAR LA DYNAMIQUE
Georges DUMONT, Parie-Paule GASCUEL, Anne VERROUST
Février 1991, 84 pages

PI 572 MULTIGRID MOTION ESTIMATION ON PYRAMIDAL REPRESENTATIONS FOR IMAGE SEQUENCE CODING
Nadia BAAZIZ, Claude LABIT
Février 1991, 48 pages

PI 573 A SURVEY OF TREE-TRANDUCTIONS
Jean-Claude RAOULT
Février 1991, 18 pages

PI 574 THE OPTIMAL ADAPTATIVE CONTROL USING RECURSIVE IDENTIFICATION
Anatolij B. JUDITSKY
Février 1991 - 26 pages

PI 575 MANUEL SIGNAL
Patricia BOURNAI, Bruno CHERON, Bernard HOUSSAIS, Paul LE
Paul LE GUERNIC
Février 1991, 84 pages

PI 576 AN INFORMATION BASED RELIABILITY PREDICTOR FOR SYSTEMS IN OPERATIONAL PHASE
Kamel SISMAIL
Février 1991 - 22 pages

PI 577 MULTISCALE STATISTICAL SIGNAL PROCESSING AND RANDOM FIELDS ON HOMOGENEOUS TREES
Albert BENVENISTE, Michèle BASSEVILLE, Ramine NIKOUKHAH, Alan S. WILLSKY, Ken C. CHOU
Mars 1991 - 18 pages

PI 578 TOWARDS A DECLARATIVE METHOD FOR 3D SCENE SKETCH MODELING
Stéphane DONIKIAN, Gérard HEGRON
Mars 1991 - 22 pages

PI 579 SYSTEMES MARKOVIENS DISCRETS STATIONNAIRES ET APPLICATIONS
Jean PELLAUMAIL
Mars 1991 - 284 pages

ISSN 0249-6399